Internet Draft
Expires May 26, 1997
File: draft-ietf-rsvp-routing-01.ps

Daniel Zappala
USC/ISI
November 1996

RSRR: A Routing Interface For RSVP

November 26, 1996

Status of Memo

This document is an Internet-Draft. Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

To learn the current status of any Internet-Draft, please check the "1id-abstracts.txt" listing contained in the Internet-Drafts Shadow Directories on ds.internic.net (US East Coast), nic.nordu.net (Europe), ftp.isi.edu (US West Coast), or munnari.oz.au (Pacific Rim).

Abstract

This memo describes a routing interface for RSVP. The interface allows RSVP to request information and services from routing in the form of an asynchronous query-reply protocol. The primary addition to this version of the document is the inclusion of extensions for shared trees appropriate to PIM and CBT. Aside from these extensions, the rest of the interface described in this document is implemented in ISI's implementation of RSVP and Xerox PARC's distribution of DVMRP.

1 Introduction

This document describes an asynchronous routing interface by which RSVP [6] may request forwarding information at each router in the network. We call this interface Routing Support for Resource Reservations (RSRR), because it may conceivably be used by other resource reservation protocols. The primary addition to this version of the document is the inclusion of extensions for shared trees appropriate to PIM [3] and CBT [1]. Aside from these extensions, the rest of the interface described in this document is implemented in ISI's implementation of RSVP and Xerox PARC's distribution of DVMRP [4].

This document is written from the perspective of multicast routing; however, the interface can also be used by RSVP to interact with a unicast routing protocol. This document elaborates on the description contained in the RSVP spec [2]. Some familiarity with RSVP is assumed.

Section 2 presents a brief overview of RSVP functionality and describes how RSVP uses route queries and route change notification. Section 3 details a general interface model that routing protocols can use to give configuration and forwarding information to RSVP. Section 4 outlines the particular queries and responses that comprise the routing interface, focusing on how RSVP uses this interface. Finally, Section 5 details the specification of the interface messages.

2 RSVP Overview

Using RSVP, sources send *Path* messages hop-by-hop to the destination (Figure 1a). Each router uses the *Path* message to create reverse-path routing state and forwards the message toward the destination. Receivers send *Resv* messages toward sources, following the reverse-path routing state (Figure 1b). Each router uses the *Resv* message to query admission control to accept or reject the embedded reservation request. Reservation messages utilize various *styles* to allow sharing among different senders. For example, the shared-explicit style targets a reservation at a list of senders, and the wildcard style targets a reservation at all upstream senders (Figure 1c).

RSVP does not use its own routing protocol; instead it uses underlying routing protocols to determine where it should carry resource reservations. In keeping with this modularity, we have designed the RSR interface, by which RSVP requests routing services from various routing protocols (Figure 2). Using this interface, RSVP may acquire routing entries to allow it to send control messages hop-by-hop, as well as request *route change notification*.

RSVP acquires multicast routing entries by sending *route queries* to the routing protocol. RSVP uses the routing entries to simulate the multicast of *PATH* messages. Normally, an IP multicast packet sent out one interface is looped back and sent out the other interfaces of a router. However, RSVP needs to send a different copy of a *PATH* message out each

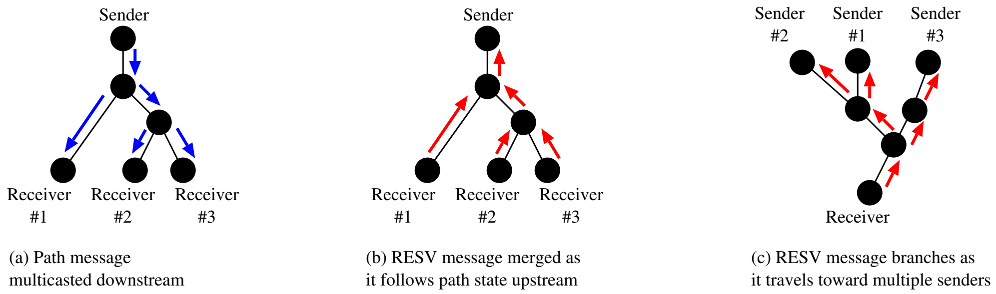

Figure 1: RSVP Overview

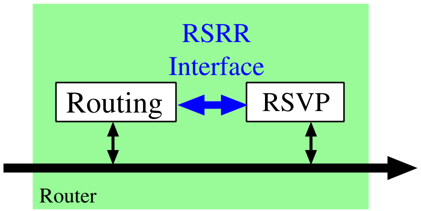

Figure 2: RSRR Interface

outgoing interface. Thus, rather than relying on IP multicast, RSVP simulates its own multicast forwarding so that it can specify a single interface to send a multicast packet, without any loop back.

When RSVP acquires routing entries, it may also ask routing to notify it when a particular route changes. RSVP uses route change notification so that it can quickly adapt its reservations to changes in the route between a source and destination. For multicast destinations, a route change consists of any local change in the multicast tree for a source-group pair, including prunes and grafts as well as routing changes due to failed or recovered links. In all of these cases, RSVP adapts to route changes by re-sending *Path* or *Resv* messages where needed. If routing can not support route change notification, then RSVP must poll routing for route entries in order to adapt to route changes.

3 RSRR Interface Abstraction

Because RSVP must learn about routing entries from a variety of different routing protocols, we have adopted portions of the DVMRP interface abstraction [4] as the means by which RSVP can communicate with all routing protocols. A routing entry for a source-destination pair consists of an incoming virtual interface (or vif) and a set of outgoing virtual interfaces. A virtual interface is simply a number that routing creates to refer to each of the interfaces (or virtual interfaces) it uses to send and receive packets on the router. Note that the implementation of the virtual interface is hidden from RSVP; whether the interface is a real interface, a pseudo-device, or a tunnel is irrelevant.

When RSVP receives a packet, it expects the forwarding engine to tell it which virtual interface the packet arrived on. This allows RSVP to suppress forwarding of packets that routing has determined have arrived via the wrong incoming virtual interface, while still allowing local processing. When RSVP sends a packet, it expects that it can tell the forwarding engine to forward the packet directly over a particular vif, without performing any of the forwarding engine's usual routing checks or lookups. Together, these functions allow RSVP to forward distinct packets hop-by-hop over each link in a unicast or multicast path between a source and destination(s).

The RSVP routing model defines a virtual interface using the following attributes:

id	a unique identifier,
threshold	a TTL threshold,
status	a bitmask status vector, and
local_addr	a local address.

RSVP uses the TTL threshold to control the scope of a control message. The use of administrative scoping (as with DVMRP) by a routing protocol will affect the forwarding information given to RSVP, but its implementation is transparent to RSVP. The status vector currently only defines whether the virtual interface has been administratively disabled.

4 RSRR Messages

Before RSVP can obtain routing entries, it must first discover which virtual interfaces (vifs) routing is using. It does this by issuing an Initial Query:

$$Initial\_Query().$$

Routing responds with an Initial Reply that includes the number of vifs and a list of vifs and their attributes as defined by the RSVP routing model:

$$Initial\_Reply(num, vif\_list).$$

Once RSVP has received the Initial Reply, it can begin requesting routing entries by sending a Route Query for a source-destination pair:

$$Route\_Query(source, destination).$$

Routing responds by sending a Route Reply that includes the incoming vif and an outgoing vif bitmask:

$$Route\_Reply(source, destination, incoming\_vif, outgoing\_vif\_bitmask).$$

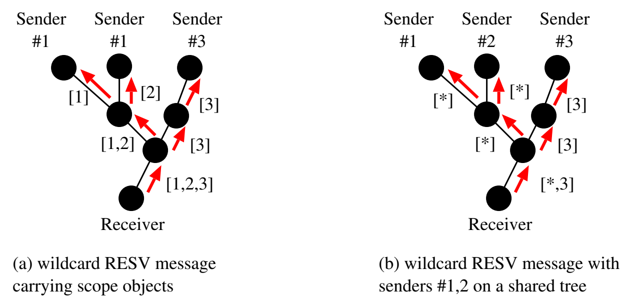

Figure 3: The Scope Object in Wildcard Reservations

RSVP can request notification of route changes by sending a Route Query with an additional notification flag:

$$Route\_Query(source, destination, notification).$$

By setting the notification flag in the query, RSVP requests that routing provide route change notification. If routing is able to provide this service, it sets a corresponding notification flag in the Route Reply, otherwise it clears the flag. If RSVP receives a Route Reply with the notification bit set, it can assume that routing will notify it – via a spontaneous Route Reply – when the routing entry for the source-destination pair changes. In the meantime, RSVP can use the routing entry indefinitely.

Routing incurs very little cost for providing route change notification; essentially it only has to tag the subset of its routing entries for which RSVP is interested in receiving notification. This amounts to keeping an extra bit for each routing entry. Since this service can be provided independently by each router, its implementation is not subject to any interoperability constraints.

A routing protocol that uses shared trees (such as PIM[3] or CBT[1]) can help RSVP to decrease the size of the SCOPE object in wildcard filter RESV messages. RSVP uses a SCOPE object, listing all upstream senders, to prevent looping of wildcard filter RESV messages [5] (Figure 3a). For shared trees, RSVP can use a SCOPE object that includes a wildcard address, greatly reducing the size of the RESV message. A RESV message with a wildcard address would follow shared tree state but never sender tree state (Figure 3b). Routing can indicate a particular sender is using a shared tree by setting the shared flag in a Route Reply:

$$Route\_Reply(source, destination, incoming\_vif, outgoing\_vif\_bitmask, shared).$$

If at some later point this sender switches to a sender-based tree, routing can send an updated Route Reply with this flag cleared.

A routing protocol that uses shared trees can also help reduce the amount of message passing between RSVP and routing. RSVP normally sends a separate Route Query for every active source in a group. For a shared tree, RSVP only needs to send one query, since all the routing entries for a given destination will be the same. Routing can indicate that *all* senders are using a shared tree by setting the all-shared flag:

$$Route\_Reply(source, destination, incoming\_vif, outgoing\_vif\_bitmask, all\_shared).$$

If, at some later point, a sender switches to a sender-based tree (i.e. with PIM), then routing can send an updated Route Reply with this flag cleared.

5 RSRR Specification

This section details the format of RSRR messages received and sent by a routing protocol.

5.1 RSRR message format

An RSRR message consists of:

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Version       | Type          | Flags         | Num           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|...                                                            |
|                                                               |
```

```
Version
        Routing Support for Resource Reservations Version.  This
        document specifies version 1 of RSRR.

Type
        This document defines four message codes for RSRR:

                1 = Initial Query
                2 = Initial Reply
                3 = Route Query
                4 = Route Reply

The rest of the message is defined separately for each RSRR code.
```

5.2 Initial Query

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Version       | Type          | Flags         | Num           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

```
Version as defined above.

Type
        1 = Initial Query

Flags, Num
        0
```

5.3 Initial Reply

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Version       | Type          | Flags         | Num           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Vif ID-1      |Vif Threshold-1| Vif Status-1                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Vif Local Address-1                                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|...                                                            |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Vif ID-N      |Vif Threshold-N| Vif Status-N                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Vif Local Address-N                                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

Version as defined above.

Type
        2 = Initial Reply

Flags
        0

Num
        Number of vifs being reported

Vif ID-N
        ID for the Nth vif

Vif Threshold-N
        The threshold ttl for the vif; an outgoing message must have a
        ttl greater than the threshold to be sent
```

Vif Status-N
A bit vector representing the vif status. Currently only the Disabled bit is defined:

```
+-+-+-+-+-+-+-+-+
|             |D|
+-+-+-+-+-+-+-+-+
```

D = 1 if vif is administratively disabled, 0 otherwise.

The rest of the field is transmitted as zeroes.

Vif Local Address-N
The local address of the physical interface corresponding to the vif

5.4 Route Query

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Version       | Type          | Flags         | Num           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Destination Address                                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Source Address                                                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Query Identifier                                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Version as defined above

Type
3 = Route Query

Flags
Currently only the Notification Bit is defined:

```
+-+-+-+-+-+-+-+-+
|             |N|
+-+-+-+-+-+-+-+-+
```

N = 1 if RSVP requests route change notification for this query, 0 otherwise.

The rest of the field is transmitted as zeroes.

Num
    0

Destination Address
    Group address being queried

Source Address
    Source address being queried

Query Identifier
    Identifier used by reservation protocol

5.5 Route Reply

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Version       | Type          | Flags         | Num           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Destination Address                                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Source Address                                                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Query Identifier                                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Incoming Vif                  |        Reserved               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Outgoing Vif Bitmask                                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Version as defined above.

Type
    4 = Route Reply

Flags
    The currently defined flags are:

```
+-+-+-+-+-+-+-+-+
|       | S |E|N|
+-+-+-+-+-+-+-+-+
```

    N is set if N is set in the corresponding route query and the

```
        router can perform route change notification for the query.
        Otherwise N is cleared.

        E is set if routing is unable to obtain routing information for
        the route query.  Otherwise E is cleared.

        S has the binary value 01 if the listed sender is using a shared
        tree, but some other senders for the same destination use sender
        trees.  S has the binary value 10 if all senders for the
        destination use shared trees.  Otherwise, S has the value 00.

        The rest of the field is transmitted as zeroes.

Destination Address
        group address of query = group address of reply

Source Address
        source address of query = source address of reply

Query Identifier
        identifier used by reservation protocol, copied from query message

Incoming Vif
        incoming Vif for (S,G) or default (S,*) if no group-specific
                state; invalid if E bit is set

Reserved
        transmitted as 0

Outgoing Vif Bitmask
        bitmask of outgoing Vifs for (S,G) or default (S,*) if no
                group-specific state; invalid if E bit is set
```

6 Acknowledgments

We would like to thank Bob Braden, Deborah Estrin, Bill Fenner, Scott Shenker, and Lixia Zhang for their help with this work.

References

[1] A. J. Ballardie, P.F. Francis, and J. Crowcroft. “Core Based Trees”. In *ACM SIG-*

COMM, August 1993.

[2] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin. "Resource ReSerVation Protocol (RSVP) - Version 1 Functional Specification". work in progress, May 1996.

[3] Stephen Deering, Deborah Estrin, Dino Farinacci, Van Jacobson, Ching-Gung Liu, and Liming Wei. An Architecture for Wide-Area Multicast Routing. In *ACM SIGCOMM*, August 1994.

[4] D. Waitzman, C. Partridge, and S. Deering. "Distance Vector Multicast Routing Protocol". RFC 1075, November 1988.

[5] Daniel Zappala. "RSVP Loop Prevention for Wildcard Reservations". Work in Progress, February 1996.

[6] Lixia Zhang, Steve Deering, Deborah Estrin, Scott Shenker, and Daniel Zappala. "RSVP: A New Resource ReSerVation Protocol". *IEEE Network*, September 1993.

Security Considerations

Security considerations are not discussed in this memo.

Author's Address

Daniel Zappala
USC Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA 90292
EMail: daniel@isi.edu